A Regularized Graph Layout Framework for Dynamic Network Visualization

Kevin S. Xu* Mark Kliger† Alfred O. Hero III‡

February 28, 2012

Abstract

Many real-world networks, including social and information networks, are dynamic structures that evolve over time. Such dynamic networks are typically visualized using a sequence of static graph layouts. In addition to providing a visual representation of the network topology at each time step, the sequence should preserve the *mental map* between layouts of consecutive time steps to allow a human to interpret the temporal evolution of the network. In this paper, we propose a framework for dynamic network visualization using *regularized graph layouts*. Regularization encourages stability of the layouts over time, thus preserving the mental map. The proposed framework involves optimizing a modified cost function that augments the cost function of a static graph layout algorithm with a *grouping penalty*, which encourages nodes to stay close to other nodes belonging to the same group, and a *temporal penalty*, which encourages smooth movements of the nodes over time. We introduce two dynamic layout algorithms under this framework, namely dynamic multidimensional scaling (DMDS) and dynamic graph Laplacian layout (DGLL), that are regularized versions of their static counterparts. We apply the proposed algorithms on several data sets to illustrate the importance of regularization for producing interpretable visualizations of dynamic networks.

1 Introduction

The study of networks has emerged as a topic of great interest in recent years, with applications in the social, computer, and life sciences, among others. Dynamic networks are of particular interest because networks observed in the real

*EECS Department, University of Michigan, Ann Arbor, MI, USA, Email: xukevin@umich.edu
†Medasense Biometrics, Ltd., Ofakim, Israel, Email: mark@medasense.com
‡EECS Department, University of Michigan, Ann Arbor, MI, USA, Email: hero@umich.edu

world often evolve over time due to the creation of new nodes and edges and the removal of old nodes and edges (Kossinets and Watts, 2006; Leskovec et al., 2007). Many developments have been made in mining dynamic networks, including finding low-rank approximations (Sun et al., 2007; Tong et al., 2008) and the detection of clusters or communities and how they evolve over time (Chi et al., 2009; Mucha et al., 2010; Xu et al., 2011a). However, the closely related task of visualizing dynamic networks has remained an open problem that has attracted attention from sociologists (Moody et al., 2005; Bender-deMoll and McFarland, 2006; Leydesdorff and Schank, 2008) and the information visualization community (Brandes and Wagner, 1997; Brandes and Corman, 2003; Erten et al., 2004; Brandes et al., 2006; Frishman and Tal, 2008) among others. Visualization is an important tool that can provide insights and intuition about networks that cannot be conveyed by summary statistics alone.

Visualizing static networks is a challenge in itself. Static networks are typically represented by graphs, which have no natural representation in a Euclidean space. Many graph layout algorithms have been developed to create aesthetically pleasing 2-D representations of graphs (Di Battista et al., 1999; Herman et al., 2000). Common layout methods for general graphs include force-directed layout (Kamada and Kawai, 1989; Fruchterman and Reingold, 1991), multidimensional scaling (MDS) (de Leeuw and Heiser, 1980; Gansner et al., 2004; Borg and Groenen, 2005) and graph Laplacian layout (GLL), also known as spectral layout (Hall, 1970; Koren, 2003).

Dynamic networks are typically represented by a time-indexed sequence of graph snapshots; thus visualizing dynamic networks in 2-D presents an additional challenge due to the temporal aspect (Branke, 2001). If one axis is used to represent time, then only a single axis remains to convey the topology of the network. While it is possible to identify certain trends from a 1-D time plot created in this manner, it is a poor representation of the network topology. Brandes and Corman (2003) presented a possible solution to this problem by creating a pseudo-3-D visualization that treats 2-D layouts of each snapshot as layers in a stack. Unfortunately, the resulting visualization is difficult to interpret. The more conventional approach is to present an animated 2-D layout that evolves over time to reflect the current snapshot (Erten et al., 2004; Moody et al., 2005; Bender-deMoll and McFarland, 2006; Brandes et al., 2006; Frishman and Tal, 2008). A challenge with this approach is to preserve the *mental map* (Misue et al., 1995) between graph snapshots so that the transition between frames in the animation can be easily interpreted by a human viewer. In particular, it is undesirable for a large number of nodes to drastically change positions between frames, which may cause the viewer to lose reference of the previous layout.

Some of the early work on dynamic network visualization simply involved creating interpolated transition layouts (Moody et al., 2005; Bender-deMoll and McFarland,

2006). While interpolation does make an animation more aesthetically pleasing, it does not constrain the successive layouts in any way to make them more interpretable. In many real networks, individual snapshots have high variance, so creating a layout for each snapshot using a static graph layout method could result in large node movements between time steps. Often, this is not due to a failure of the static graph layout algorithm but simply a consequence of the cost function it is attempting to optimize, which does not consider any other snapshots. When dealing with dynamic networks, better performance can be obtained by using *regularized* methods that consider the historical snapshots in addition to the current snapshot. Such an approach has been used to develop regularized clustering algorithms for dynamic networks, also known as evolutionary clustering algorithms (Chi et al., 2009; Mucha et al., 2010; Xu et al., 2011a), that outperform traditional static clustering algorithms in the dynamic setting. In the context of dynamic network visualization, regularization encourages layouts to be stable over time, thus preserving the mental map between snapshots. The concept of regularization has been employed in many problems in statistics and machine learning, including ridge regression (Hoerl and Kennard, 1970), also known as Tikhonov regularization, the LASSO (Tibshirani, 1996), and penalized matrix decomposition (Witten et al., 2009). It is often used to introduce additional information or constraints and to give preference to solutions with certain desirable properties such as sparsity, smoothness, or in this paper, dynamic stability in order to preserve the mental map.

We introduce a framework for dynamic network visualization using *regularized graph layouts*. The framework is designed to generate layouts in the *on-line* setting where only present and past snapshots are available, although it can also be used in the *off-line* setting where access to future snapshots is also available. It involves optimizing a modified cost function that augments the cost function of a static graph layout algorithm with two penalties:

1. A grouping penalty, which discourages nodes from deviating too far from other nodes belonging to the same group;

2. A temporal penalty, which discourages nodes from deviating too far from their previous positions.

Groups could correspond to a priori knowledge, such as participant affiliations in social networks. If no a priori group knowledge is available, groups can be learned from the network using, for example, the aforementioned evolutionary clustering algorithms. The grouping penalty keeps group members close together in the sequence of layouts, which helps to preserve the mental map because nodes belonging to the same group often evolve over time in a similar fashion. The tempo-

ral penalty helps to preserve the mental map by discouraging large node movements that may cause a human to lose reference of previous node positions, such as multiple nodes moving unnecessarily from one side of the layout to the opposite side. Using the proposed framework, we develop two dynamic layout algorithms, *dynamic multidimensional scaling* (DMDS) and *dynamic graph Laplacian layout* (DGLL), that are regularized versions of MDS and GLL, respectively.

The idea of preserving temporal stability in dynamic graph layouts has been proposed in previous work (Brandes and Wagner, 1997; Erten et al., 2004; Baur and Schank, 2008; Frishman and Tal, 2008). The idea of placing nodes belonging to the same group together in a layout has also appeared in previous work (Wang and Miyamoto, 1996; Huang and Eades, 1998; Costa and Hero III, 2005), albeit in the static graph setting. To the best of our knowledge, this is the first work that presents a framework for dynamic network visualization[1] that incorporates *both grouping and temporal regularization*. The algorithms DMDS and DGLL are also unique in that they add both grouping and temporal regularization to MDS and GLL, respectively. Existing methods for temporal regularization in MDS (Baur and Schank, 2008) and grouping regularization in GLL (Costa and Hero III, 2005) are subsumed by DMDS and DGLL, respectively. The methods for grouping regularization in MDS and temporal regularization in GLL used in this paper are novel. We apply the proposed algorithms on a selection of dynamic network data sets to demonstrate the importance of both grouping and temporal regularization in creating interpretable visualizations.

2 Background

We begin by specifying the notation we shall use in this paper. Time-indexed quantities are indicated using square brackets, e.g. $X[t]$. We represent a dynamic network by a discrete-time sequence of graph snapshots. Each snapshot is represented by a graph adjacency matrix $W[t]$ where $w_{ij}[t]$ denotes the weight of the edge between nodes i and j at time t (chosen to be 1 for unweighted graphs), and $w_{ij}[t] = 0$ if no edge is present. We assume all graphs are undirected, so that $w_{ij}[t] = w_{ji}[t]$. For simplicity of notation, we typically drop the time index for all quantities at time step t, i.e. W is assumed to denote $W[t]$.

We refer to a graph layout by a matrix $X \in \mathbb{R}^{n \times s}$, where n denotes the number of nodes present at time t, and each row $\mathbf{x}_{(i)}$ corresponds to the s-dimensional position of node i. We are most interested in 2-D visualization ($s = 2$), although the proposed methods can also be applied to other values of s. The ith column of X is denoted by $\mathbf{x}_i$, and the individual entries by x_{ij}. The superscript in $\mathbf{x}_i^{(h)}$ is

[1] A preliminary version of this work can be found in (Xu et al., 2011b).

used to denote the value of $\mathbf{x}_i$ at iteration h of an algorithm. The norm operator $\|\cdot\|$ refers to the l_2-norm, and $\mathrm{tr}(\cdot)$ denotes the matrix trace operator. We denote the all ones column vector by $\mathbf{1}$.

We now summarize the static graph layout methods of multidimensional scaling and graph Laplacian layout, which operate on a single graph snapshot. We develop regularized versions of these methods for dynamic networks in Section 3.

2.1 Multidimensional scaling

Multidimensional scaling (MDS) is a family of statistical methods that aim to find an optimal layout $X \in \mathbb{R}^{n\times s}$ such that the distance between $\mathbf{x}_{(i)}$ and $\mathbf{x}_{(j)}$ for all $i \neq j$ is as close as possible to a desired distance δ_{ij}. There are a variety of different cost functions and associated algorithms for MDS; we refer interested readers to Borg and Groenen (2005). Here we describe the cost function known as stress and its associated majorization algorithm. The stress of a layout X is given by

$$\mathrm{Stress}(X) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} v_{ij}\left(\delta_{ij} - \|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}\|\right)^2, \tag{1}$$

where $v_{ij} \geq 0$ denotes the weight or importance of maintaining the desired distance δ_{ij}. We refer to the matrix $V = [v_{ij}]$ as the *MDS weight* matrix to avoid confusion with the graph adjacency matrix W, which is also sometimes referred to as a weight matrix. In order to use stress MDS for graph layout, the graph adjacency matrix W is first converted into a desired distance matrix $\Delta = [\delta_{ij}]$. This is done by defining a distance metric over the graph and calculating distances between all pairs of nodes. The distance between two nodes i and j is typically taken to be the length of the shortest path between the nodes (Gansner et al., 2004). For weighted graphs, it is assumed that the edge weights denote dissimilarities; if the edge weights instead denote similarities, they must first be converted into dissimilarities before computing Δ. The MDS weights v_{ij} play a crucial role in the aesthetics of the layout. The commonly used Kamada-Kawai (KK) force-directed layout (Kamada and Kawai, 1989) is a special case of stress MDS where the weights are chosen to be $v_{ij} = \delta_{ij}^{-2}$ for $i \neq j$ and $v_{ii} = 0$.

The objective of stress MDS is to find a layout X that minimizes (1). (1) can be decomposed into

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} v_{ij}\delta_{ij}^2 + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} v_{ij}\|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}\|^2 - \sum_{i=1}^{n}\sum_{j=1}^{n} v_{ij}\delta_{ij}\|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}\|. \tag{2}$$

Note that the first term of (2) is independent of X. The second term of (2) can be

written as $\operatorname{tr}(X^T RX)$ where the $n \times n$ matrix R is given by

$$r_{ij} = \begin{cases} -v_{ij} & i \neq j, \\ \sum_{k \neq i} v_{ik} & i = j. \end{cases} \tag{3}$$

$\operatorname{tr}(X^T RX)$ is quadratic and convex in X and is easily optimized.

The third term of (2) cannot be written as a quadratic form. However, it can be optimized by an iterative majorization method known as "scaling by majorizing a complicated function" (SMACOF) (de Leeuw and Heiser, 1980; Borg and Groenen, 2005). This non-quadratic term is iteratively majorized, and the resulting upper bound for the stress is then optimized by differentiation. For a matrix $Z \in \mathbb{R}^{n \times s}$, define the matrix-valued function $S(Z)$ by

$$s_{ij}(Z) = \begin{cases} -v_{ij}\delta_{ij} / \|\mathbf{z}_{(i)} - \mathbf{z}_{(j)}\| & i \neq j, \\ -\sum_{k \neq i} s_{ik}(Z) & i = j. \end{cases} \tag{4}$$

Then, it is shown in (Gansner et al., 2004; Borg and Groenen, 2005) that

$$-\operatorname{tr}(X^T S(Z)Z) \geq -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} v_{ij}\delta_{ij} \|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}\|$$

so that an upper bound for the stress is

$$\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} v_{ij}\delta_{ij}^2 + \operatorname{tr}(X^T RX) - 2\operatorname{tr}(X^T S(Z)Z). \tag{5}$$

By setting the derivative of (5) with respect to X to 0, the minimizer of the upper bound is found to be the solution to the equation $RX = S(Z)Z$.

The algorithm for optimizing stress is iterative. Let $X^{(0)}$ denote an initial layout. Then at each iteration h, solve

$$R\,\mathbf{x}_a^{(h)} = S\left(X^{(h-1)}\right)\mathbf{x}_a^{(h-1)} \tag{6}$$

for $\mathbf{x}_a^{(h)}$ for each $a = 1, \ldots, s$. (6) can be solved using a standard linear equation solver. Note that R is rank-deficient; this is a consequence of the stress function being translation-invariant (Gansner et al., 2004). The translation-invariance can be removed by fixing the location of one point, e.g. by setting $\mathbf{x}_{(1)} = 0$, removing the first row and column of R, and removing the first row of $S\big(X^{(h-1)}\big)X^{(h-1)}$ (Gansner et al., 2004). (6) can then be solved efficiently using Cholesky factorization. The iteration can be terminated when

$$\frac{\operatorname{Stress}\left(X^{(h-1)}\right) - \operatorname{Stress}\left(X^{(h)}\right)}{\operatorname{Stress}\left(X^{(h-1)}\right)} < \epsilon, \tag{7}$$

where ϵ is the convergence tolerance of the iterative process.

2.2 Graph Laplacian layout

Graph Laplacian layout (GLL) methods optimize a quadratic function associated with the graph Laplacian matrix (Koren, 2003), which we call the GLL energy. The graph Laplacian is obtained from the adjacency matrix by $L = D - W$, where D is the diagonal matrix of node degrees defined by $d_{ii} = \sum_{j=1}^{n} w_{ij}$. For weighted graphs, GLL assumes that the weights correspond to similarities between nodes, rather than dissimilarities as in MDS. GLL is also referred to as "spectral layout" because the optimal solution involves the eigenvectors of the Laplacian, as we will show. The GLL energy function is defined by

$$\mathrm{Energy}(X) = \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} \|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}\|^2. \tag{8}$$

It is easily shown that $\mathrm{Energy}(X) = \mathrm{tr}(X^T L X)$. The GLL problem can be formulated as (Hall, 1970; Koren, 2003):

$$\min_{X} \quad \mathrm{tr}(X^T L X) \tag{9}$$

$$\text{subject to} \quad X^T X = nI \tag{10}$$

$$X^T \mathbf{1} = \mathbf{0}. \tag{11}$$

From (8), it can be seen that minimizing the GLL energy function aims to make edge lengths short by placing nodes connected by heavy edges close together. (11) removes the trivial solution $\mathbf{x}_a = \mathbf{1}$, which places all nodes at the same location in one dimension. It can also be viewed as removing a degree of freedom in the layout due to translation invariance (Belkin and Niyogi, 2003) by setting the mean of $\mathbf{x}_a$ to 0 for all a. Since $\mathbf{x}_a$ has zero-mean, $(\mathbf{x}_a^T \mathbf{x}_a)/n$ corresponds to the variance or scatter of the layout in dimension a. Thus (10) constrains the layout to have unit variance in each dimension and zero covariance between dimensions so that each dimension of the layout provides as much additional information as possible. Moreover, one can see that (10) differs slightly from the usual constraint $X^T X = I$ (Belkin and Niyogi, 2003; Koren, 2003), which constrains the layout to have variance $1/n$ in each dimension. In the dynamic network setting where n can vary over time, this is undesirable because the layout would change scale between time steps if the number of nodes changes.

By a generalization of the Rayleigh-Ritz theorem (Lütkepohl, 1997), an optimal solution to the GLL problem is given by $X^* = \sqrt{n}[\mathbf{v}_2, \mathbf{v}_3, \ldots, \mathbf{v}_{s+1}]$, where $\mathbf{v}_i$ denotes the eigenvector corresponding to the ith smallest eigenvalue of L. Note that $\mathbf{v}_1 = (1/\sqrt{n})\mathbf{1}$ is excluded because it violates the zero-mean constraint (11). Using the property that $\mathrm{tr}(ABC) = \mathrm{tr}(CAB)$, the cost function (9) is easily

shown to be invariant to rotation and reflection, so X^*R is also an optimal solution for any $R^TR = I$.

In practice, it has been found that using degree-normalized eigenvectors often results in more aesthetically pleasing layouts (Belkin and Niyogi, 2003; Koren, 2003). The degree-normalized layout problem differs only in that the dot product in each of the constraints is replaced with the degree-weighted dot product, resulting in the following optimization problem:

$$\begin{aligned} \min_X \quad & \operatorname{tr}(X^T L X) \\ \text{subject to} \quad & \operatorname{tr}(X^T D X) = \operatorname{tr}(D) I \\ & X^T D \mathbf{1} = \mathbf{0}. \end{aligned}$$

The optimal solution is given by $X^* = \sqrt{\operatorname{tr}(D)}\,[\mathbf{u}_2, \mathbf{u}_3, \ldots, \mathbf{u}_{s+1}]$ or any rotation or reflection of X^*, where $\mathbf{u}_i$ denotes the generalized eigenvector corresponding to the ith smallest generalized eigenvalue of (L, D). Again, $\mathbf{u}_1 = \left(1/\sqrt{\operatorname{tr}(D)}\right)\mathbf{1}$ is excluded because it violates the zero-mean constraint. A discussion on the merits of the degree normalization can be found in (Koren, 2003).

3 Regularized layout methods

3.1 Regularization framework

The aforementioned static layout methods can be applied snapshot-by-snapshot to create a visualization of a dynamic network; however, the resulting visualization is often difficult to interpret, especially if there are large node movements between time steps. We propose a regularized layout framework that uses a *modified cost* function, defined by

$$\mathcal{C}_{\text{modified}} = \mathcal{C}_{\text{static}} + \alpha \mathcal{C}_{\text{grouping}} + \beta \mathcal{C}_{\text{temporal}}. \tag{12}$$

The static cost $\mathcal{C}_{\text{static}}$ corresponds to the cost function optimized by the static layout algorithm. For example, in MDS, it is the stress function defined in (1), and in GLL, it is the energy defined in (8). The grouping cost $\mathcal{C}_{\text{grouping}}$ is chosen to discourage nodes from deviating too far from other group members; α controls the importance of the grouping cost, so we refer to $\alpha\mathcal{C}_{\text{grouping}}$ as the grouping penalty. Similarly, the temporal cost $\mathcal{C}_{\text{temporal}}$ is chosen to discourage nodes from deviating too far from their previous positions; β controls the importance of the temporal cost, so we refer to $\beta\mathcal{C}_{\text{temporal}}$ as the temporal penalty. We propose quadratic forms for these penalties, similar to ridge regression (Hoerl and Kennard, 1970).

Let k denote the number of groups. Define the group membership by an $n \times k$ matrix C where

$$c_{il} = \begin{cases} 1 & \text{node } i \text{ is in group } l \text{ at time step } t, \\ 0 & \text{otherwise.} \end{cases}$$

We introduce grouping regularization by adding group representatives, which also get mapped to an s-dimensional position, stored in the matrix $Y \in \mathbb{R}^{k \times s}$. The proposed grouping cost is given by

$$\mathcal{C}_{\text{grouping}}(X, Y) = \sum_{l=1}^{k} \sum_{i=1}^{n} c_{il} \|\mathbf{x}_{(i)} - \mathbf{y}_{(l)}\|^2, \tag{13}$$

where $\mathbf{y}_{(l)}$ denotes the position of the lth representative. Notice that the grouping cost is decreased by moving $\mathbf{y}_{(l)}$ and $\mathbf{x}_{(i)}$ towards each other if node i is in group l. As a result, nodes belonging to the same group will be placed closer together than in a layout without grouping regularization. Notice also that we do not require knowledge of the group membership of every node. Nodes with unknown group memberships correspond to all-zero rows in C and are not subject to any grouping penalty.

We introduce temporal regularization on nodes present at both time steps t and $t-1$ by discouraging node positions from deviating significantly from their previous positions. This idea is often referred to in the information visualization literature as maintaining *dynamic stability* of the layouts and is often used to achieve the goal of preserving the mental map. Define the diagonal matrix E by

$$e_{ii} = \begin{cases} 1 & \text{node } i \text{ was present at time step } t-1, \\ 0 & \text{otherwise.} \end{cases}$$

The proposed temporal cost is then given by

$$\mathcal{C}_{\text{temporal}}(X, X[t-1]) = \sum_{i=1}^{n} e_{ii} \|\mathbf{x}_{(i)} - \mathbf{x}_{(i)}[t-1]\|^2. \tag{14}$$

The temporal cost is decreased by moving $\mathbf{x}_{(i)}$ towards $\mathbf{x}_{(i)}[t-1]$, but unlike in the grouping cost, $\mathbf{x}_{(i)}[t-1]$ is fixed because it was assigned at the previous time step. Thus the previous node position acts as an anchor for the current node position.

We note that the temporal cost measures only the stability of the layouts over time and is not necessarily a measure of goodness-of-fit with regard to the dynamic network. For example, if there is a sudden change in the network topology, an extremely low temporal cost may be undesirable because it could prevent the layout

from adequately adapting to reflect this change. Thus one must consider the trade-off of adaptation rate versus stability when choosing the temporal regularization parameter.

Next we demonstrate how the grouping and temporal penalties can be introduced into MDS and GLL as examples of the proposed regularization framework.

3.2 Dynamic multidimensional scaling

The dynamic multidimensional scaling (DMDS) modified cost is given by the modified stress function

$$\begin{aligned}\mathrm{MStress}(X,Y) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} v_{ij}\big(\delta_{ij} - \|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}\|\big)^2 \\ + \alpha\sum_{l=1}^{k}\sum_{i=1}^{n} c_{il}\|\mathbf{x}_{(i)} - \mathbf{y}_{(k)}\|^2 + \beta\sum_{i=1}^{n} e_{ii}\|\mathbf{x}_{(i)} - \mathbf{x}_{(i)}[t-1]\|^2.\end{aligned} \tag{15}$$

The first term of (15) is the usual MDS stress function, while the second term corresponds to the grouping penalty, and the third term corresponds to the temporal penalty. The constants α and β are the grouping and temporal regularization parameters, respectively.

To optimize (15), we begin by re-writing the first two terms into a single term. Define the augmented MDS weight matrix by

$$\tilde{V} = \begin{bmatrix} V & \alpha C \\ \alpha C^T & 0 \end{bmatrix}, \tag{16}$$

where the zero corresponds to an appropriately sized all-zero matrix. Similarly, define the $(n+k)\times(n+k)$ augmented desired distance matrix $\tilde{\Delta}$ by filling the added rows and columns with zeros, i.e.

$$\tilde{\Delta} = \begin{bmatrix} \Delta & 0 \\ 0 & 0 \end{bmatrix} \tag{17}$$

Let

$$\tilde{X} = \begin{bmatrix} X \\ Y \end{bmatrix} \tag{18}$$

denote the positions of the both the nodes and the group representatives. Then, the first two terms of (15) can be written as

$$\frac{1}{2}\sum_{i=1}^{n+k}\sum_{j=1}^{n+k} \tilde{v}_{ij}\left(\tilde{\delta}_{ij} - \|\tilde{\mathbf{x}}_{(i)} - \tilde{\mathbf{x}}_{(j)}\|\right)^2,$$

which has the same form as the usual stress defined in (1). The third term in (15) can be written as a quadratic function of $\tilde{X}$, namely

$$\beta\left[\operatorname{tr}\left(\tilde{X}^T\tilde{E}\tilde{X}\right) - 2\operatorname{tr}\left(\tilde{X}^T\tilde{E}\tilde{X}[t-1]\right) + \operatorname{tr}\left(\tilde{X}^T[t-1]\tilde{E}\tilde{X}[t-1]\right)\right],$$

where the $(n+k)\times(n+k)$ matrix $\tilde{E}$ and the $(n+k)\times s$ matrix $\tilde{X}[t-1]$ are constructed by zero-filling as in the definition of $\tilde{\Delta}$.

Following the derivation in Section 2.1, for any $(n+k)\times s$ matrix Z, (15) can be majorized by

$$\begin{aligned}\frac{1}{2}\sum_{i=1}^{n+k}\sum_{j=1}^{n+k}\tilde{v}_{ij}\tilde{\delta}_{ij}^2 + \operatorname{tr}(\tilde{X}^T\tilde{R}\tilde{X}) - 2\operatorname{tr}(\tilde{X}^T\tilde{S}(Z)Z) + \beta\Big[\operatorname{tr}(\tilde{X}^T\tilde{E}\tilde{X}) \\ - 2\operatorname{tr}(\tilde{X}^T\tilde{E}\tilde{X}[t-1]) + \operatorname{tr}(\tilde{X}^T[t-1]\tilde{E}\tilde{X}[t-1])\Big],\end{aligned} \tag{19}$$

where $\tilde{R}$ and $\tilde{S}$ are defined by substituting the augmented matrices $\tilde{V}$ and $\tilde{\Delta}$ for V and Δ, respectively, in (3) and (4). (19) is quadratic and convex in X so the minimizer of the upper bound is found by setting the derivative of (19) to 0, resulting in the equation

$$\big(\tilde{R}+\beta\tilde{E}\big)\tilde{X} = \tilde{S}\big(Z\big)Z + \beta\tilde{E}\tilde{X}[t-1].$$

This can again be solved sequentially over each dimension. As in Section 2.1, we solve this iteratively using the previous iteration as the majorizer, i.e. at iteration h, solve

$$\big(\tilde{R}+\beta\tilde{E}\big)\tilde{\mathbf{x}}_a^{(h)} = \tilde{S}\left(\tilde{X}^{(h-1)}\right)\tilde{\mathbf{x}}_a^{(h-1)} + \beta\tilde{E}\tilde{\mathbf{x}}_a[t-1]. \tag{20}$$

for $\tilde{\mathbf{x}}_a^{(h)}$ for each $a = 1,\ldots,s$. The process is iterated until the convergence criterion (7) is attained. The first iterate can be taken to be simply the previous layout $\tilde{\mathbf{x}}_a[t-1]$. Unlike in ordinary MDS, the system of linear equations in (20) has a unique solution provided that at least a single node was present at time step $t-1$, because $\tilde{R}+\beta\tilde{E}$ has full rank in this case.

Pseudocode for the DMDS algorithm for $t = 1, 2, \ldots$ is shown in Fig. 1, where $\mathrm{shortest\_paths}(\cdot)$ computes the matrix of shortest paths between all pairs of nodes, and $\mathrm{MDS\_weights}(\cdot)$ computes the MDS weight matrix. (20) can be solved by performing a Cholesky factorization on $(\tilde{R}+\beta\tilde{E})$ followed by back substitution. At the initial time step ($t = 0$), there are no previous node positions to initialize with, so a random initialization is used. Also, the position of one node should be fixed before solving (20) due to the translation-invariance discussed in Section 2.1. The time complexity of the algorithm at all subsequent time steps is dominated by the $O(n^3)$ complexity of the Cholesky factorization, assuming $k \ll n$, but

for $t = 1, 2, \ldots$ **do**
  $\Delta \leftarrow \text{shortest\_paths}(W)$
  $V \leftarrow \text{MDS\_weights}(\Delta)$
  Construct $\tilde{V}$ and $\tilde{\Delta}$ using (16) and (17), respectively
  Construct $\tilde{R}$ by substituting $\tilde{V}$ for V in (3)
  $h \leftarrow 0$
  $\tilde{X}^{(0)} \leftarrow \tilde{X}[t-1]$
  repeat
    $h \leftarrow h + 1$
    Construct $\tilde{S}\big(\tilde{X}^{(h-1)}\big)$ by substituting $\tilde{V}$, $\tilde{\Delta}$, and $\tilde{X}^{(h-1)}$ for V, Δ, and Z, respectively, in (4)
    for $a = 1, \ldots, s$ **do**
      Solve $\big(\tilde{R} + \beta\tilde{E}\big)\tilde{\mathbf{x}}_a^{(h)} = \tilde{S}\big(\tilde{X}^{(h-1)}\big)\tilde{\mathbf{x}}_a^{(h-1)} + \beta\tilde{E}\tilde{\mathbf{x}}_a[t-1]$ for $\tilde{\mathbf{x}}_a^{(h)}$
    end for
  until $\big[\,\text{MStress}\big(\tilde{X}^{(h-1)}\big) - \text{MStress}\big(\tilde{X}^{(h)}\big)\big] / \text{MStress}\big(\tilde{X}^{(h-1)}\big) < \epsilon$
  $\tilde{X} \leftarrow \tilde{X}^{(h)}$
end for

Figure 1: Pseudocode for the DMDS algorithm.

the factorization only needs to be computed at the initial iteration ($h = 1$). All subsequent iterations require only matrix-vector products and back substitution and thus have $O(n^2)$ complexity.

3.3 Dynamic graph Laplacian layout

The dynamic graph Laplacian layout (DGLL) modified cost is given by the modified energy function

$$\begin{aligned}\text{MEnergy}(X, Y) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}\|\mathbf{x}_{(i)} - \mathbf{x}_{(j)}\|^2 \\ + \alpha\sum_{l=1}^{k}\sum_{i=1}^{n} c_{il}\|\mathbf{x}_{(i)} - \mathbf{y}_{(l)}\|^2 + \beta\sum_{i=1}^{n} e_{ii}\|\mathbf{x}_{(i)} - \mathbf{x}_{(i)}[t-1]\|^2.\end{aligned} \tag{21}$$

Like with DMDS, the first term of (21) is the usual GLL energy function, while the second term corresponds to the grouping penalty, and the third term corresponds to the temporal penalty. Again, the parameters α and β correspond to the grouping and temporal regularization parameters, respectively.

We first re-write (21) in a more compact form using the graph Laplacian. Define the augmented adjacency matrix by

$$\tilde{W} = \begin{bmatrix} W & \alpha C \\ \alpha C^T & 0 \end{bmatrix}. \tag{22}$$

Notice that the group representatives have been added as nodes to the graph, with edges between each node and its associated representative of weight α. Define the augmented degree matrix by $\tilde{D}$ by $\tilde{d}_{ii} = \sum_{j=1}^{n+k} \tilde{w}_{ij}$, and the augmented graph Laplacian by $\tilde{L} = \tilde{D} - \tilde{W}$. The first two terms of (21) can thus be written as $\operatorname{tr}(\tilde{X}^T \tilde{L} \tilde{X})$, where $\tilde{X}$ is as defined in (18). The third term of (21) can be written as

$$\beta \left[\operatorname{tr}(\tilde{X}^T \tilde{E} \tilde{X}) - 2 \operatorname{tr}(\tilde{X}^T \tilde{E} \tilde{X}[t-1]) + \operatorname{tr}(\tilde{X}^T[t-1] \tilde{E} \tilde{X}[t-1]) \right], \tag{23}$$

where $\tilde{E}$ is zero-filled as described in Section 3.2. The final term in (23) is independent of $\tilde{X}$ and is henceforth dropped from the modified cost.

We now consider the constraints, which differ depending on whether the layout is degree-normalized, as discussed in Section 2.2. We derive the constraints for the degree-normalized layout; the equivalent constraints for the unnormalized layout can simply be obtained by replacing $\tilde{D}$ with the identity matrix in the derivation. First we note that, due to the temporal regularization, the optimal layout is no longer translation-invariant, so we can remove the zero-mean constraint. As a result, the variance and orthogonality constraints become more complicated because we need to subtract the mean. Denote the degree-weighted mean in dimension a by

$$\tilde{\mu}_a = \frac{1}{\sum_{i=1}^{n+k} \tilde{d}_{ii}} \sum_{i=1}^{n+k} \tilde{d}_{ii} \tilde{x}_{ia}.$$

Then the degree-weighted covariance between the ath and bth dimensions is given by

$$\begin{aligned} \operatorname{cov}(\tilde{\mathbf{x}}_a, \tilde{\mathbf{x}}_b) &= \frac{1}{\sum_{i=1}^{n+k} \tilde{d}_{ii}} \sum_{i=1}^{n+k} \tilde{d}_{ii} (\tilde{x}_{ia} - \tilde{\mu}_a)(\tilde{x}_{ib} - \tilde{\mu}_b) \\ &= \frac{1}{\sum_{i=1}^{n+k} \tilde{d}_{ii}} \sum_{i=1}^{n+k} \tilde{d}_{ii} \tilde{x}_{ia} \tilde{x}_{ib} - \frac{1}{\left(\sum_{i=1}^{n+k} \tilde{d}_{ii} \right)^2} \left(\sum_{i=1}^{n+k} \tilde{d}_{ii} \tilde{x}_{ia} \right) \left(\sum_{i=1}^{n+k} \tilde{d}_{ii} \tilde{x}_{ib} \right) \\ &= \frac{\tilde{\mathbf{x}}_a^T M \tilde{\mathbf{x}}_b}{\operatorname{tr}(\tilde{D})}, \end{aligned}$$

where M is the centering matrix defined by

$$M = \tilde{D} - \frac{\tilde{D}\mathbf{1}\mathbf{1}^T\tilde{D}}{\mathrm{tr}(\tilde{D})}. \tag{24}$$

Combining the modified cost function with the modified constraints, the normalized DGLL problem is as follows:

$$\min_{\tilde{X}} \quad \mathrm{tr}(\tilde{X}^T\tilde{L}\tilde{X}) + \beta\left[\mathrm{tr}(\tilde{X}^T\tilde{E}\tilde{X}) - 2\tilde{X}^T\tilde{E}\tilde{X}[t-1])\right] \tag{25}$$

$$\text{subject to} \quad \mathrm{tr}(\tilde{X}^T M\tilde{X}) = \mathrm{tr}(\tilde{D})I. \tag{26}$$

Again, the unnormalized problem can be obtained by replacing $\tilde{D}$ with the identity matrix in (24) and (26). Note that (25) contains a linear term in $\tilde{X}$. Hence the optimal solution cannot be obtained using scaled generalized eigenvectors as in the static GLL problem. (25) can be solved using standard algorithms for constrained nonlinear optimization (Bazaraa et al., 2006). The cost function and constraints consist only of linear and quadratic terms, so the gradient and Hessian are easily computed in closed form (see Appendix A). Unfortunately, the problem is not convex due to the equality constraints; thus a good initialization is important. The natural choice is to initialize using the previous layout $\tilde{X}^{(0)} = \tilde{X}[t-1]$. To avoid getting stuck in poor local minima, one could use multiple restarts with random initialization.

Pseudocode for the DGLL algorithm for $t = 1, 2, \ldots$ is shown in Fig. 2. We use the interior-point algorithm of Byrd et al. (1999) to solve (25). We find in practice that random restarts are not necessary unless β is extremely small because the temporal regularization penalizes solutions that deviate too far from the previous layout. For other choices of β, we find that the interior-point algorithm indeed converges to the global minimum when initialized using the previous layout. The most time-consuming operation in solving (25) consists of a Cholesky factorization, which must be updated at each iteration. At the initial time step ($t = 0$), there are no previous node positions, and hence, no linear term in (25), so the layout is obtained using scaled generalized eigenvectors, as described in Section 2.2. The time complexity at all subsequent time steps is dominated by the $O(n^3)$ complexity of the Cholesky factorization.

3.4 Discussion

We chose to demonstrate the proposed framework with MDS and GLL; however, it is also applicable to other graph layout methods, such as the Fruchterman-Reingold method of force-directed layout (Fruchterman and Reingold, 1991). Since the static

for $t = 1, 2, \ldots$ **do**
  Construct $\tilde{W}$ using (22) and its corresponding Laplacian $\tilde{L} = \tilde{D} - \tilde{W}$
  Construct the centering matrix M using (24)
  $\tilde{X}^{(0)} \leftarrow \tilde{X}[t-1]$
  Solve (25) using the forms for ∇f, g, H, and J in Appendix A
  for $r = 1 \rightarrow max\_restarts$ **do** {if multiple random restarts are necessary}
    Randomly assign $\tilde{X}^{(0)}$
    Solve (25) using the forms for ∇f, g, H, and J in Appendix A
  end for
  $\tilde{X} \leftarrow$ best solution to (25) over all initializations
end for

Figure 2: Pseudocode for the DGLL algorithm.

cost functions of MDS and GLL encourage different appearances, the same is true of DMDS and DGLL. Ultimately, the decision of which type of layout to use depends on the type of network and user preferences. Kamada-Kawai MDS layouts are often preferred in 2-D because they discourage nodes from overlapping due to the large MDS weights assigned to maintaining small desired distances. On the other hand, if a 1-D layout is desired, so that the entire sequence can be plotted as a time series, node overlap is a lesser concern. For such applications, DGLL may be a better choice.

Another decision that needs to be made by the user is the choice of the parameters α and β, which can be tuned as desired to create a meaningful animation. Unlike in supervised learning tasks such as classification, there is no ground truth in visualization so the selection of parameters in layout methods is typically done in an ad-hoc fashion. Furthermore, multiple layouts created by differing choices of parameters could be useful for visualizing different portions of the network or yielding different insights (Witten and Tibshirani, 2011). This is particularly true of the grouping regularization parameter α. When a high value of α is used, nodes belonging to the same group are placed much closer together than nodes belonging to different groups. The resulting visualization emphasizes node movements between groups (for nodes that change group between time steps) while sacrificing the quality of the node movements within groups. On the other hand, when a low value of α is used, node movements within groups are more clearly visible, but node movements between groups are more difficult to see. We explore the effect of changing parameters on the resulting animation in several experiments in Section 5.

4 Related work

The regularized graph layout framework proposed in this paper utilizes a *grouping* penalty that places nodes belonging to the same group together and a *temporal* penalty that places nodes near their positions at neighboring time steps. Node grouping and temporal stability in the context of graph layout have previously been studied independently of each other. We summarize relevant contributions to both of these areas.

4.1 Node grouping

Several grouping techniques for static graph layout have been previously been proposed. Given a partition of a graph into groups, Wang and Miyamoto (1996) propose a modified force-directed layout that considers three types of forces: intra-forces, inter-forces, and meta-forces. Intra-forces and inter-forces denote forces between nodes in the same group and nodes in different groups, respectively. Meta-forces correspond to forces between groups; nodes in the same group are subject to identical meta-forces. By decreasing the strength of inter-forces and increasing the strength of meta-forces, nodes belonging to the same group get positioned closer to each other in the layout.

Huang and Eades (1998) developed a system called DA-TU for visualizing groups in large static graphs. It also utilizes a modified force-directed layout with inter- and intra-forces. However, rather than using meta-forces, DA-TU adds a virtual node for each group with a virtual force between each virtual node and each node in its group. Notice that the virtual nodes are identical to the group representatives in our proposed framework; however, the use of virtual forces to achieve grouping regularization differs from the squared Euclidean distance grouping cost we propose. In addition, DA-TU was designed for visualizing static graphs at many scales rather than visualizing dynamic graphs, so it does not contain a temporal stability penalty.

Grouping techniques have been applied in the field of supervised dimensionality reduction, which is very closely related to graph layout. The objective of dimensionality reduction (DR) is to find a mapping $\phi : \mathbb{R}^p \to \mathbb{R}^s$, $p > s$ from a high-dimensional space to a lower-dimensional space while preserving many of the characteristics of the data representation in the high-dimensional space (Lee and Verleysen, 2007). For example, MDS is a DR method that attempts to preserve pairwise distances between data points. In the supervised DR setting, one also has a priori knowledge of the grouping structure of the data. Supervised DR methods pose the additional constraint that data points within the same group should be closer together in the low-dimensional space than points in separate groups. Notice that

this is the same grouping constraint we pose in our regularized layout framework.

Witten and Tibshirani (2011) proposed a supervised version of MDS (SMDS) that optimizes the following cost function over X:

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\delta_{ij}-\|\mathbf{x}_{(i)}-\mathbf{x}_{(j)}\|)^2+\alpha\sum_{i,j:y_j>y_i}(y_j-y_i)\sum_{a=1}^{s}\left(\frac{\delta_{ij}}{\sqrt{s}}-(x_{ja}-x_{ia})\right)^2, \tag{27}$$

where y_i is an ordinal value denoting the group membership of data point i. Notice that the first term in (27) is the ordinary MDS stress with $v_{ij} = 1$ for all i, j, while the second term provides the grouping regularization. α controls the trade-off between the two terms. The key difference between the SMDS grouping penalty and the DMDS grouping penalty proposed in this paper is in the way groups are treated. SMDS assumes that groups are labeled with an ordinal value that allows them to be ranked, and the form of the grouping penalty in (27) does indeed tend to rank groups in $\mathbb{R}^s$ by encouraging $x_{ja} > x_{ia}, a = 1, \ldots, s$ for all $i, j : y_j > y_i$. On the other hand, our proposed grouping penalty treats group labels as categorical. It does not rank groups in $\mathbb{R}^s$ but simply pulls nodes belonging to the group together.

Another related method for supervised DR is classification constrained dimensionality reduction (CCDR) (Costa and Hero III, 2005), which is a supervised version of Laplacian eigenmaps (Belkin and Niyogi, 2003). CCDR optimizes the following cost function over (X, Y):

$$\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|\mathbf{x}_{(i)}-\mathbf{x}_{(j)}\|^2+\alpha\sum_{l=1}^{k}\sum_{i=1}^{n}c_{il}\|\mathbf{x}_{(i)}-\mathbf{y}_{(l)}\|^2.$$

Notice that this cost function is simply the sum of the GLL energy and the DGLL grouping penalty. Indeed, DGLL can be viewed as an extension of CCDR to time-varying data. The CCDR solution is given by the matrix of generalized eigenvectors $\tilde{U} = [\tilde{\mathbf{u}}_2, \ldots, \tilde{\mathbf{u}}_{s+1}]$ of $(\tilde{L}, \tilde{D})$, where $\tilde{L}$ and $\tilde{D}$ are as defined in Section 3.3. Although the addition of the temporal regularization due to the anchoring presence of the previous layout $X[t-1]$ prevents the DGLL problem from being solved using generalized eigenvectors, it discourages large node movements between time steps in order to better preserve the mental map.

4.2 Temporal stability

There have been many previous studies on the problem of laying out dynamic networks while preserving stability between time snapshots. Moody et al. (2005) proposed to generate dynamic layouts using a static layout method such as Kamada-Kawai MDS and to initialize at each time step using the layout generated at the

previous time step. The idea of this approach is to anchor the nodes initially so that the entire layout does not get rotated. The anchoring differs from the temporal regularization proposed in this paper, which penalizes changes in node positions over time and can be thought of as anchoring all iterations rather than just the initial iteration. The approach of Moody et al. (2005) is implemented in the social network visualization software SoNIA (Bender-deMoll and McFarland, 2011). Experimentally, we find that solely anchoring the initialization is insufficient at preventing drastic node movements over time (see Section 5 for examples).

To enforce stability in layouts at consecutive time steps, Brandes and Wagner (1997) proposed a probabilistic framework for dynamic network layout where the objective is to choose the layout at a particular time step with maximum posterior probability given the previous layout. The layout with maximum posterior probability can be found by maximizing the product of the prior probability and likelihood of the layout. Similar to our proposed framework, the probabilistic framework is applicable to a wide class of graph layout methods. The prior probability of a layout is taken to be an (appropriately normalized) inverse exponential function of the static cost of the layout. Brandes and Wagner (1997) proposed several different types of likelihood functions corresponding to different notions of temporal stability. One such notion is to demand stability of node positions in consecutive layouts. For this notion of stability, the authors propose the likelihood for each node to be a spherical Gaussian distribution centered at the node's position in the layout; the likelihood of the layout is then taken to be the product of the likelihoods of all nodes present in consecutive time steps. Thus the posterior probability can be written as (up to a normalizing constant),

$$\exp\left\{-\left(\mathcal{C}_{\text{static}}+\frac{\sum_{i=1}^{n} e_{ii}\|\vec{x}_{(i)}-\vec{x}_{(i)}[t-1]\|^2}{2\sigma^2}\right)\right\}, \tag{28}$$

where σ is a scaling parameter for the amplitude of node movements. Notice that by taking the logarithm of (28), one obtains the same form as our proposed regularized framework (12), excluding the grouping cost, with $\beta = 1/(2\sigma^2)$. The layout that maximizes the logarithm of (28) is the same layout that maximizes posterior probability; thus, under this notion of stability, there is an equivalence between the probabilistic framework of Brandes and Wagner (1997) and the temporal regularization framework proposed in this paper.

Other methods for preserving temporal stability in dynamic layouts tend to be specific to a particular layout method. Baur and Schank (2008) proposed a temporally regularized MDS algorithm that uses the following localized update rule at

each iteration h for each node i at each time step t:

$$x_{ia}^{(h)} = \frac{\tilde{x}_{ia}^{(h-1)} + \beta(e_{ii}x_{ia}[t-1] + f_{ii}x_{ia}[t+1])}{\sum_{j\neq i} v_{ij} + \beta(e_{ii} + f_{ii})}, \tag{29}$$

where

$$\tilde{x}_{ia}^{(h-1)} = \sum_{j\neq i} v_{ij} \left(x_{ja}^{(h-1)} + \delta_{ij} \frac{x_{ia}^{(h-1)} - x_{ja}^{(h-1)}}{\|\mathbf{x}_{(i)}^{(h-1)} - \mathbf{x}_{(j)}^{(h-1)}\|} \right),$$

and F is the diagonal matrix defined by

$$f_{ii} = \begin{cases} 1 & \text{node } i \text{ is present at time step } t+1, \\ 0 & \text{otherwise.} \end{cases}$$

This algorithm is implemented in the social network analysis and visualization software Visone (Brandes and Wagner, 2004; Visone) and was used in (Leydesdorff and Schank, 2008) for visualizing similarities in journal content over time. (29) is an off-line update because it uses both the node positions at time steps $t-1$ and $t+1$ to compute the node position at time step t, whereas the methods we propose, including DMDS, are on-line methods that use only current and past data. (29) can be modified into an on-line update by removing the terms involving f_{ii}. It was shown in (Baur and Schank, 2008) that the localized update of (29) optimizes the sum of the $\mathrm{MStress}$ function in (15) over all t with $k = 0$, i.e. without a grouping penalty. Likewise, the on-line modification optimizes the $\mathrm{MStress}$ function at a single time step with $k = 0$. Hence the proposed DMDS layout method can be viewed as an on-line modification of the method of (Baur and Schank, 2008) with the addition of a grouping penalty.

When it comes to GLL, to the best of our knowledge, there is no prior work that explicitly penalizes large node movements. Brandes et al. (2006) suggest two approaches for layout of dynamic networks using GLL. The first is to simply compute a static layout for each graph snapshot using the spectral method described in Section 2.2. The eigenvectors are calculated using power iteration (Trefethen and Bau III, 1997) initialized with the layout at the previous time step, similar to the approach of Moody et al. (2005) for MDS. As previously mentioned, this does not ensure layout stability. The second approach is to compute a spectral layout of a smoothed graph Laplacian matrix $\lambda L[t-1] + (1-\lambda)L[t]$. A similar approach has been used for clustering dynamic networks (Chi et al., 2009; Xu et al., 2011a). This approach incorporates information from two graph snapshots and should perform better than the first approach, but it also does not explicitly constrain the layout at time t from deviating too far from the layout at time $t-1$. We henceforth refer to this approach

as the BFP method and compare its performance to our proposed DGLL algorithm in Section 5.

Other methods for layout of dynamic networks have also been proposed (Erten et al., 2004; Frishman and Tal, 2008). TGRIP (Erten et al., 2004) is a modified force-directed layout method with added edges between vertices present at multiple time steps. The user-selected weights of these added edges control the amount of temporal regularization in the layouts. The method of Frishman and Tal (2008) is also a modified force-directed layout. It is an on-line method that uses pinning weights to previous node positions to achieve temporal regularization and a GPU-based implementation to reduce run-time. The emphasis in both methods is on improving scalability to deal with extremely large networks by coarsening graphs to compute an initial layout then applying local refinements to improve the quality of the layout. As a result, they are applicable to much larger networks than the $O(n^3)$ methods proposed in this paper. However, these methods do not incorporate any sort of grouping regularization to discourage nodes from deviating too far from other nodes in the same group.

5 Experiments

We demonstrate the proposed framework by applying DMDS and DGLL on a simulated data set and two real data sets. Several snapshots of the resulting visualizations are presented. The full, animated visualizations over all time steps can be found on the supporting website (Xu et al., 2012).

In the second experiment, we do not have a priori group knowledge. Hence we learn the groups using the AFFECT evolutionary spectral clustering algorithm (Xu et al., 2011a), summarized in Appendix B. In the other two experiments, we do have a priori group knowledge. We compute layouts both using the known groups and the groups learned by clustering. We also compute layouts using several existing methods for comparison. DMDS is compared to the method of Moody et al. (2005) used in SoNIA (Bender-deMoll and McFarland, 2011) and the method of Baur and Schank (2008) used in Visone (Brandes and Wagner, 2004; Visone). The SoNIA layouts are created by anchoring the initial node positions, as described in Section 4.2. To provide a fair comparison with DMDS and SoNIA, which are on-line methods, we use the on-line modification of the Visone method described in Section 4.2. DGLL is compared to the CCDR method of Costa and Hero III (2005), the dynamic spectral layout method of Brandes et al. (2006) (referred to here as BFP), and the standard spectral GLL solution (Koren, 2003).

Summary statistics from the experiments are presented in Tables 1 and 2 for the MDS- and GLL-based methods, respectively, and are discussed in the individual

Experiment	Algorithm	MDS stress	Centroid cost	Temporal cost	Iterations
SBM	DMDS (known)	0.160 ± 0.000	$\mathbf{0.257 \pm 0.001}$	$\mathbf{0.262 \pm 0.001}$	$\mathbf{45.6 \pm 0.3}$
	DMDS (learned)	0.160 ± 0.000	0.308 ± 0.003	0.293 ± 0.002	46.9 ± 0.4
	Visone	0.157 ± 0.000	0.434 ± 0.003	0.340 ± 0.002	51.0 ± 0.4
	SoNIA	$\mathbf{0.132 \pm 0.000}$	0.623 ± 0.004	1.271 ± 0.010	112.9 ± 1.0
Newcomb	DMDS (learned)	0.136	**0.656**	**0.089**	**13.8**
	Visone	0.107	1.275	0.125	16.9
	SoNIA	**0.065**	1.611	1.368	68.5
MIT	DMDS (known)	0.154	**1.334**	**0.207**	**37.7**
	DMDS (learned)	0.154	1.491	0.241	38.6
	Visone	0.142	1.900	0.290	45.8
	SoNIA	**0.092**	2.637	3.384	108.3

Table 1: Mean costs of MDS-based layouts ($\pm$ standard error for SBM simulation experiment). The smallest quantity (within one standard error) in each column for each experiment is bolded. DMDS results using both a priori known groups (when available) and groups learned by clustering are shown.

Experiment	Algorithm	GLL energy	Centroid cost	Temporal cost
SBM	DGLL (known)	0.658 ± 0.003	$\mathbf{0.294 \pm 0.002}$	$\mathbf{0.732 \pm 0.007}$
	DGLL (learned)	0.655 ± 0.003	0.403 ± 0.005	0.858 ± 0.009
	CCDR	0.630 ± 0.003	0.414 ± 0.003	2.678 ± 0.025
	BFP	0.636 ± 0.003	0.637 ± 0.006	2.269 ± 0.027
	Spectral	$\mathbf{0.605 \pm 0.003}$	0.945 ± 0.007	3.179 ± 0.022
Newcomb	DGLL (learned)	0.820	1.325	**0.379**
	CCDR	0.786	1.334	1.352
	BFP	0.783	**1.321**	0.793
	Spectral	**0.761**	1.373	1.425
MIT	DGLL (known)	0.130	**1.228**	**0.263**
	DGLL (learned)	0.128	1.357	0.313
	CCDR	0.099	1.300	1.692
	BFP	0.104	1.471	1.897
	Spectral	**0.090**	1.660	2.478

Table 2: Mean costs of GLL-based layouts ($\pm$ standard error for SBM simulation experiment). The smallest quantity (within one standard error) in each column for each experiment is bolded. DGLL results using both a priori known groups (when available) and groups learned by clustering are shown.

subsections. The KK choice of MDS weights is used for all of the MDS-based methods, and degree-normalized layout is used for all of the GLL-based methods.

We define three measures of layout quality: static cost, centroid cost, and temporal cost. The *static cost* measures how well the current layout coordinates fit the current graph snapshot. It is the cost function that would be optimized by the static graph layout algorithm, either MDS or GLL. The static cost for the MDS-based methods is taken to be the static MDS stress defined in (1). The static cost for the GLL-based methods is the GLL energy defined in (8).

The *centroid cost* is the sum of squared distances between each node and its group centroid, which is also the cost function of the well-known method of k-means clustering. It is used to measure how close nodes are to members of their group[2]. When prior knowledge of the groups is available, we calculate the centroid cost with respect to the known groups, even for the layouts where groups are learned by clustering. When prior knowledge is not available, we calculate the centroid cost with respect to the learned groups.

The *temporal cost* (14) is the sum of squared distances between node positions in layouts at consecutive time steps. It is often used to quantify how well the mental map is preserved over time (Brandes and Wagner, 1997; Baur and Schank, 2008; Frishman and Tal, 2008). As mentioned in Section 3.1, the temporal cost should only be interpreted as a measure of stability and not a measure of temporal goodness-of-fit.

The costs displayed are appropriately normalized (either by the number of nodes or pairs of nodes, depending on the quantity) so they are comparable across different data sets. For the MDS-based methods, we also compare the number of iterations required for convergence to a tolerance of $\epsilon = 10^{-4}$. For the BFP method, the parameter λ lies on a different scale from the parameters α and β for the other methods. To ensure a fair comparison, we choose λ to minimize $\mathcal{C}_{\text{static}} + \alpha\mathcal{C}_{\text{centroid}} + \beta\mathcal{C}_{\text{temporal}}$, where α and β are chosen to be the same parameters used for the other methods.

From Tables 1 and 2, one can see that DMDS and DGLL have lower centroid and temporal costs than the competing methods in all but one instance due to the incorporation of both grouping and temporal regularization. The BFP method slightly outperforms DGLL in terms of centroid cost in the Newcomb experiment; however, BFP performs comparatively poorly in temporal cost. As expected, the lower centroid and temporal costs for DMDS and DGLL are achieved by choosing node positions with a higher static cost. Notice also that DMDS requires significantly less iterations to converge than SoNIA, which employs no regularization

[2]Note that we cannot simply use the grouping cost (13) because it is not defined for methods that do not incorporate grouping regularization.

at all, and slightly less than Visone, which employs only temporal regularization. This is an added benefit of regularization. The results for each experiment will be discussed in greater detail in the following.

5.1 Stochastic block model

In this experiment, we generate simulated networks using a stochastic block model (SBM) (Holland et al., 1983). An SBM creates networks with k groups, where nodes in a group are stochastically equivalent, i.e. the probability of forming an edge between nodes i and j is dependent only on the groups to which i and j belong. An SBM is completely specified by the set of probabilities $\{p_{cd};\, c = 1, \ldots, k;\, d = c, c+1, \ldots, k\}$, which represent the probability of forming an edge between any particular node in group c and any particular node in group d.

We generate 20 independent samples from a 30-node 4-group SBM with parameters $p_{ii} = 0.6$ and $p_{ij} = 0.2$ for all $i \neq j$. Each sample corresponds to a graph snapshot at a single time step. The group memberships are randomly assigned at the initial time step and remain unchanged up to $t = 9$. At $t = 10$, $1/4$ of the nodes are randomly re-assigned to different groups to simulate a change in the network structure. The group memberships are then held constant until the last time step. We create layouts of the network using parameters $\alpha = \beta = 1$.

In Fig. 3, we plot the variation over time of the static, centroid, and temporal costs of the MDS-based methods. The costs are averaged over 100 simulation runs. As expected, the static cost is higher for the regularized layouts than for the unregularized SoNIA layout. The grouping regularization in DMDS results in lower centroid cost as expected. When the groups are learned by clustering, the centroid cost is slightly higher than with the known groups, but still much lower than that of Visone and SoNIA. Although Visone has only temporal regularization, notice that it also has a lower centroid cost than SoNIA. This is because the SBM parameters are held constant from time steps 0 to 9 and from time steps 10 to 19, so that the group structure can be partially revealed by temporal regularization alone once enough time samples have been collected. The temporal regularization of both DMDS and Visone results in a significantly lower temporal cost than SoNIA. The grouping regularization of DMDS also decreases the temporal cost slightly compared to Visone. An added benefit of the regularization in DMDS is the significant reduction in the number of iterations required for the MDS algorithm to converge, as shown in Table 1. On average, DMDS required less than half as many iterations as SoNIA, and slightly less than Visone.

In Fig. 4, we plot the variation over time of the static, centroid, and temporal costs of the GLL-based methods. Similar to the MDS-based methods, the static cost is higher for the regularized layouts, but the centroid and temporal costs are

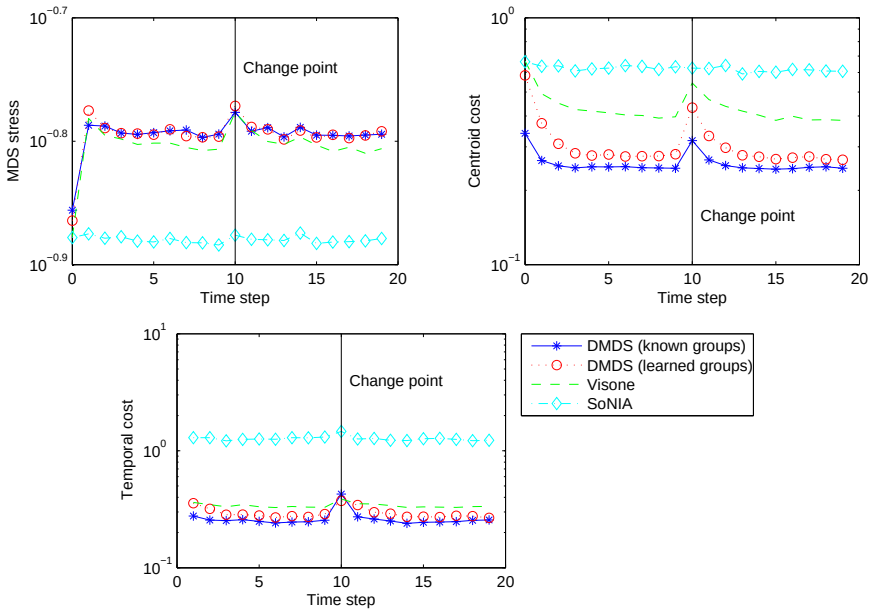

Figure 3: Costs of MDS-based layouts in the SBM experiment at each time step. The DMDS layouts have the lowest centroid and temporal costs but also the highest MDS stress due to the regularization.

much lower. Notice that only DGLL is able to generate layouts with low temporal cost. The grouping regularization in CCDR reduces the centroid cost but only slightly improves the temporal cost. The BFP method, which combines Laplacian matrices from two time steps, performs better than the standard spectral method both in centroid and temporal cost, but is worse than DGLL in both.

Notice from Figs. 3 and 4 that the centroid and temporal costs of DMDS and DGLL increase at $t = 10$, reflecting the presence of the change in network structure. Such an increase is beneficial for two reasons. First, it indicates that the grouping and temporal penalties are not so strong that they mask changes in network structure, which would be undesirable. Second, it suggests that the centroid and temporal costs can be used to filter the sequence of networks for important events, such as change points, which can be useful for exploratory analysis of dynamic networks over long periods of time.

We demonstrate the effect of varying the regularization parameters in DMDS and DGLL, respectively, in Figs. 5 and 6. We generate layouts using 10 choices each of α and β, uniformly distributed on a log scale between 0.1 and 10. The observations are similar for both DMDS and DGLL. As expected, the temporal

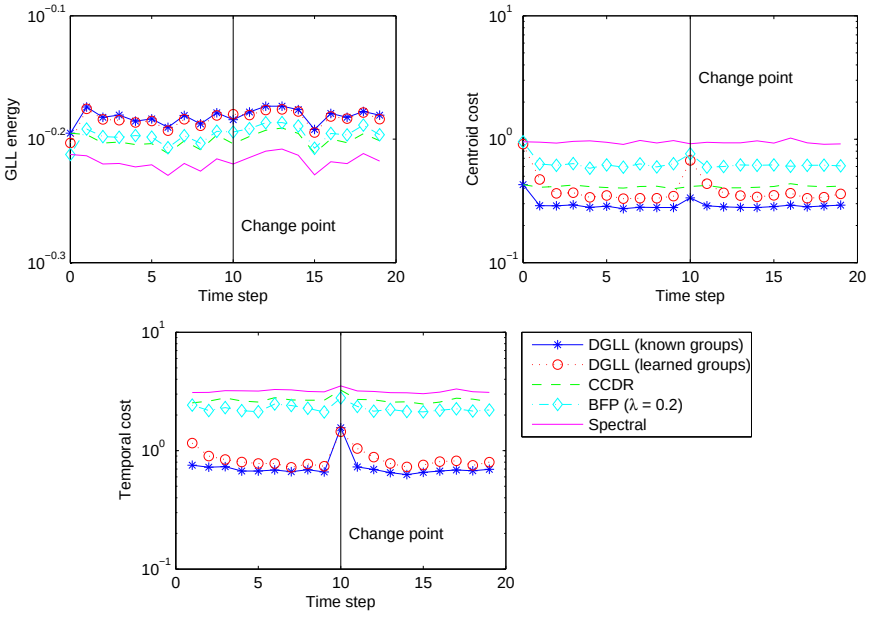

Figure 4: Costs of GLL-based layouts in the SBM experiment at each time step. The DGLL layouts have the lowest centroid and temporal costs but also the highest GLL energy due to the regularization.

cost decreases for increasing β. For low values of β, increasing α also decreases the temporal cost. This is a sensible result because nodes can move significantly over time but must remain close to the group representative, which lowers the temporal cost. The result is slightly different when it comes to the centroid cost. As expected, increasing α decreases centroid cost. For low values of α, increasing β also decreases centroid cost to a point, but a very high β may actually increase centroid cost, especially in DMDS. This is also a sensible result because a very high β places too much weight on the initial time step and prevents nodes from moving towards their group representative at future time steps.

From this experiment we can see that there is a coupled effect between grouping and temporal regularization, and that a combination of both can sometimes result in better performance with respect to both centroid and temporal cost. However, it is important to note that this is not always true. For example, if a node changes group between two time steps, then the two penalties can oppose each other, with the temporal penalty attempting to pull the node towards its previous position and the grouping penalty attempting to pull the node towards its current representative, which could be quite far from the node's previous position. This is

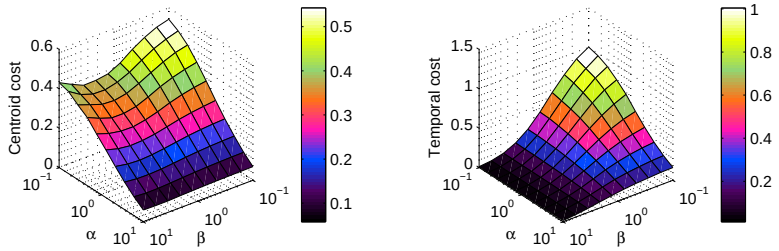

Figure 5: Mean centroid and temporal costs of DMDS layouts in the SBM experiment as functions of α and β. The centroid and temporal costs decrease as α and β are increased, respectively; however, α also affects the temporal cost, and β also affects the centroid cost.

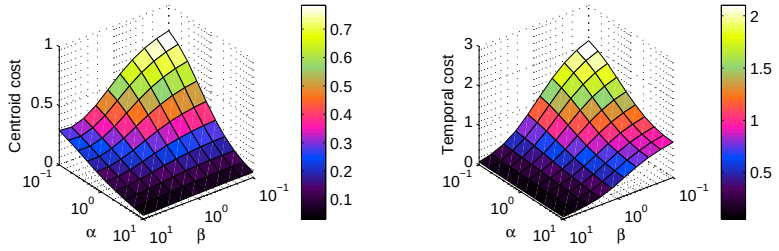

Figure 6: Mean centroid and temporal costs of DGLL layouts in the SBM experiment as functions of α and β. The behavior of both costs as functions of α and β is similar to their behavior in DMDS.

another reason for the increase in both centroid and temporal costs at $t = 10$, when the group structure is altered, in Figs. 3 and 4.

5.2 Newcomb's fraternity

This data set was collected by Nordlie and Newcomb (Nordlie, 1958; Newcomb, 1961) as part of an experiment on interpersonal relations. It has been examined in several previous studies including (Moody et al., 2005; Bender-deMoll and McFarland, 2006). 17 incoming male transfer students at the University of Michigan were housed together in fraternity housing. Each week, the participants ranked their preference of each of the other individuals in the house, in private, from 1 to 16. Data was collected over 15 weeks in a semester, with one week of data missing during week 9, corresponding to Fall break.

We process the rank data in the same manner as Moody et al. (2005) and

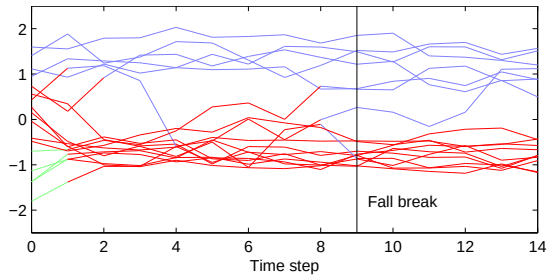

Figure 7: Time plots of 1-D DGLL layouts of Newcomb's fraternity, colored by learned groups. Node positions in the layout are relatively stable over time unless nodes are changing group.

Bender-deMoll and McFarland (2006) to give a fair comparison with SoNIA. Graph snapshots are created by connecting each participant to his 4 most preferred students with weights from 4 decreasing to 1 corresponding to the most preferred to the 4th most preferred student. The graph is converted to an undirected graph by taking the edge weight between i and j to be the larger of the directed edge weights. The weights are converted into dissimilarities for the MDS-based methods by dividing each similarity weight by the maximum similarity of 4. No group information is known a priori, so the group structure is learned using the AFFECT clustering algorithm.

In Fig. 7, we show a time plot of 1-D layouts created using DGLL, where the color of a line segment between time steps t and $t+1$ denotes the group membership of the node at time step t, and the location of the endpoints correspond to the node's position in the layouts at time steps t and $t+1$. The regularization parameters are chosen to be $\alpha = \beta = 1$. While a 1-D layout does a poor job of conveying the topology of the network, some temporal trends can be seen. For example, two mostly stable groups form after several weeks, but two nodes appear to switch from the red to the blue group around Fall break.

In Figs. 8-10, we present a comparison of the first four snapshots from the layouts created using DMDS, Visone, and SoNIA, respectively. In both figures, the top row corresponds to the layouts, and the bottom row illustrates the node movements of each node over time. In the plots on the bottom row, each node is drawn twice: once at its current position at time t and once at its previous position at time $t-1$. An edge connects these two positions; the length of the edge indicates how far a node has moved between time steps $t-1$ and t.

At $t = 0$, the red and green groups are mixed together in the Visone and SoNIA layouts, while they are easily distinguished in the DMDS layout due to the grouping regularization. Furthermore, the node movements over time in the So-

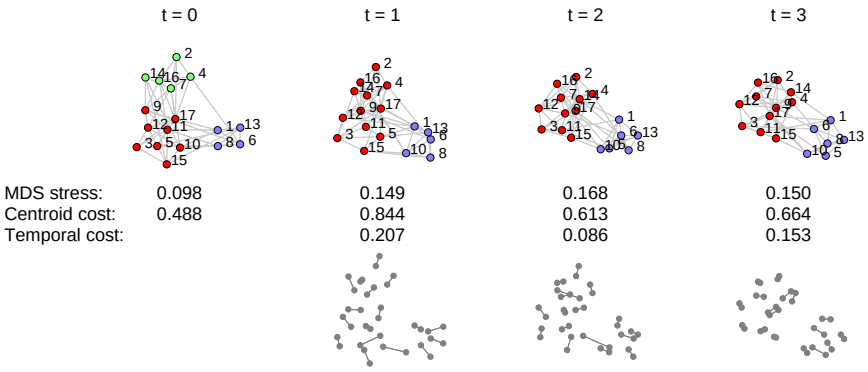

Figure 8: Layouts of Newcomb's fraternity at four time steps (top row) generated using proposed DMDS algorithm and node movements between layouts (bottom row). The groups remain well-separated.

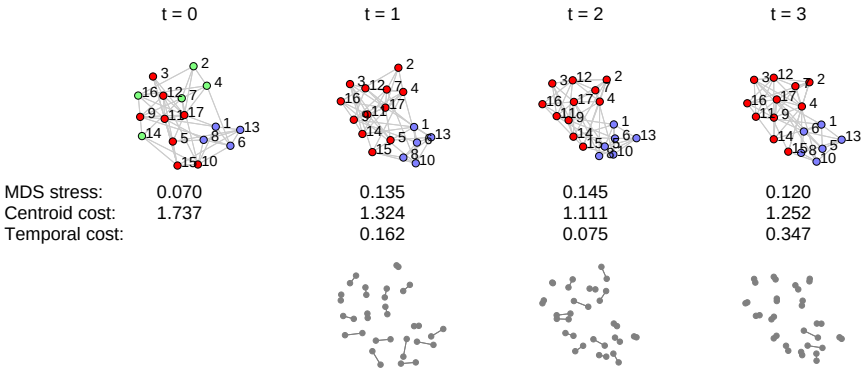

Figure 9: Layouts of Newcomb's fraternity at four time steps (top row) using the Visone algorithm and node movements between layouts (bottom row). The groups are not as well-separated as in the DMDS layouts.

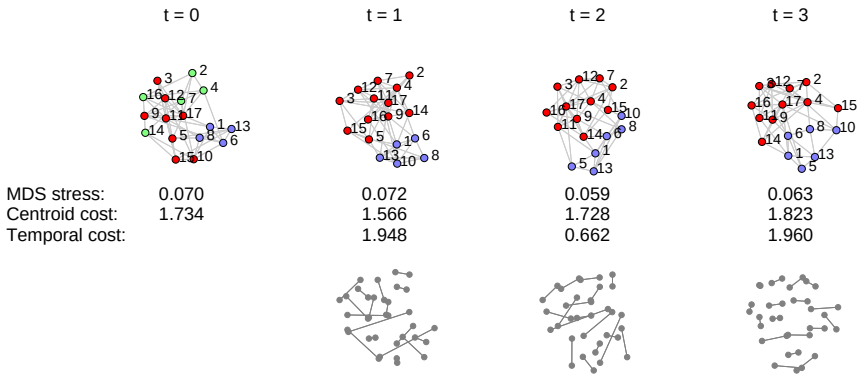

Figure 10: Layouts of Newcomb's fraternity at four time steps (top row) using the SoNIA algorithm and node movements between layouts (bottom row). There is excessive node movement, and the groups are not as well-separated as in the DMDS layouts.

NIA layouts are much more extreme than in the DMDS layouts. The excessive node movement is even more visible by comparing the DMDS and SoNIA animations on the supporting website (Xu et al., 2012). The excessive node movement is reflected in the substantially higher temporal cost of SoNIA compared to DMDS, as shown in Table 1. DMDS also outperforms Visone in both mean centroid and temporal cost, although the improvement in temporal cost is smaller than compared to SoNIA because Visone also employs temporal regularization. Finally, the mean number of iterations required for the SoNIA layout to converge is almost four times that of DMDS, so DMDS presents significant computational savings in addition to better preservation of the mental map.

For the GLL-based layouts, which can be found on the supporting website (Xu et al., 2012), there is not much difference in terms of the centroid cost. Notice from Table 2, that the mean centroid cost differs by only 4% between the best (BFP) and worst (spectral). This is not surprising, as the groups are learned using evolutionary spectral clustering, which is closely related to GLL. However, DGLL performs significantly better than the other methods in terms of temporal cost. The temporal smoothing of the Laplacian for BFP helps to lower the temporal cost slightly compared to the layouts without any temporal regularization, but it is not as effective as the temporal regularization in DGLL.

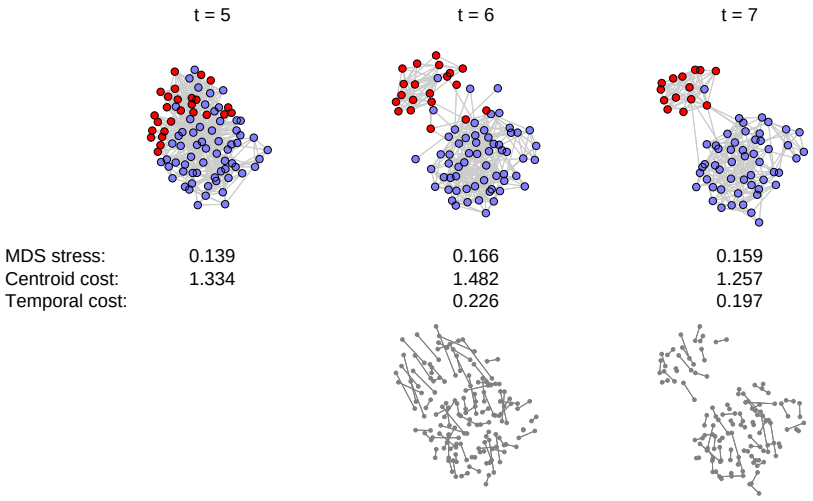

Figure 11: DMDS layouts of MIT Reality Mining data at four time steps using the known groups (top row) and node movements between layouts (bottom row). Blue nodes denote colleagues working in the same building, and red nodes denote incoming students. The incoming students separate from the others after the first week of classes ($t = 5$).

5.3 MIT Reality Mining

The MIT Reality Mining data set (Eagle et al., 2009) was collected as part of an experiment on inferring social networks by using cell phones as sensors. 94 students and staff at MIT were given access to smart phones that were monitored over two semesters. The phones were equipped with Bluetooth sensors, and each phone recorded the Media Access Control addresses of nearby Bluetooth devices at five-minute intervals. Using this proximity data, we construct a sequence of graph snapshots where each participant is connected to the 5 participants he or she was in highest proximity to during a time step. We divide the data into time steps of one week, resulting in 46 time steps between August 2004 and June 2005. From the MIT academic calendar, we know the dates of important events such as the beginning and end of school terms. We also know that 26 of the participants were incoming students at the university's business school, while the rest were colleagues working in the same building. These affiliations are used as the known groups.

The DMDS layouts at three time steps computed using the using the known

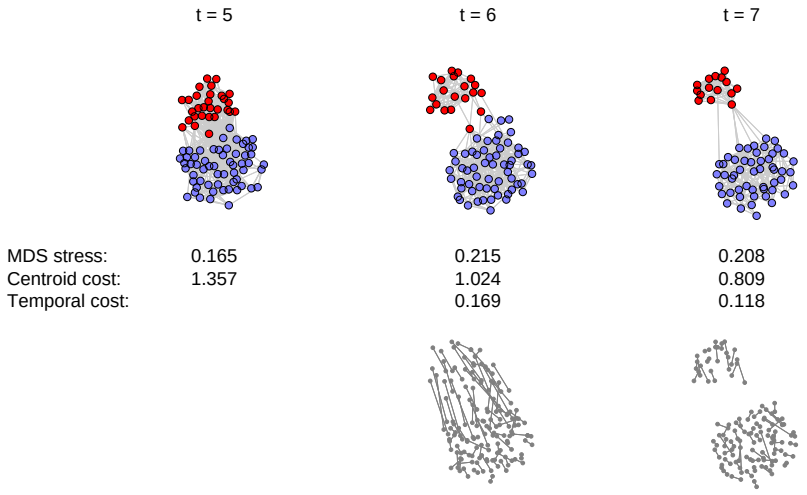

Figure 12: DMDS layouts of MIT Reality Mining data at four time steps with $\alpha = 5, \beta = 3$ using groups learned by clustering (top row) and node movements between layouts (bottom row). Colors correspond to learned groups. There is a lot of node movement between groups but very little movement within groups, resulting in high MDS stress.

groups with $\alpha = 1, \beta = 3$ are shown in Fig. 11. A higher value of β is chosen compared to the previous experiments in order to create more stable layouts due to the higher number of nodes. Node labels are not displayed to reduce clutter in the figure. We encourage readers to view the animation on the supporting website (Xu et al., 2012) to get a better idea of the temporal evolution of this network. $t = 5$ corresponds to the first week of classes. Notice that the two groups are slightly overlapped at this time step. As time progresses, the group of incoming students separates quite clearly from the colleagues working in the same building. This result suggests that the incoming students are spending more time in proximity with each other than with the remaining participants, which one would expect as the students gain familiarity with each other as the semester unfolds.

The same observation can be made from the DMDS layouts computed using the groups learned by the AFFECT clustering algorithm. Initially at $t = 5$, the separation between groups is not clear so many nodes are not correctly classified, but at subsequent time steps when the separation is clearer, almost all of the nodes are correctly classified. Between time steps 5 and 6 many nodes switch groups.

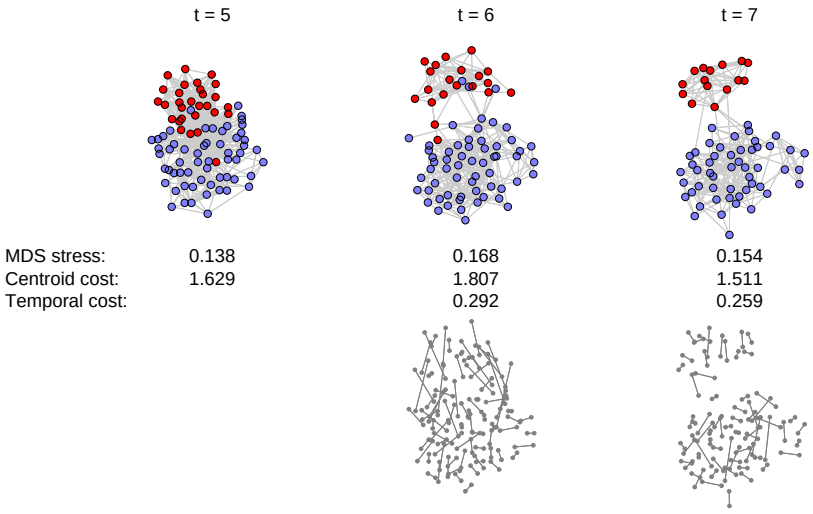

Figure 13: DMDS layouts of MIT Reality Mining data at four time steps with $\alpha = 1/5, \beta = 3$ using groups learned by clustering (top row) and node movements between layouts (bottom row). Colors correspond to learned groups. There is more movement within groups, resulting in lower MDS stress, but it is more difficult to identify movement between groups.

This can be seen in Fig. 12, where the colors correspond to the learned groups rather than the known groups. These layouts are created using $\alpha = 5, \beta = 3$; the high value of α emphasizes the node movements between groups while sacrificing the quality of movements within groups, as discussed in Section 3.4. Notice that the groups are very compact and well-separated, so that nodes switching from one group to another have large movements between layouts. Compare these layouts to those shown in Fig. 13, which are created using $\alpha = 1/5, \beta = 3$. The lower value of α better shows movements within groups, but the large changes in groups between time steps 5 and 6 are not as obvious as in Fig. 12. Both layouts are useful and provide different insights into the network dynamics; however, the observation of the incoming students separating from the other participants is evident in both visualizations.

The benefits of the regularization can be seen once again from the statistics in Tables 1 and 2. With group information provided, the DMDS and DGLL layouts have lower mean centroid and temporal costs compared to all competing layouts. Without group information, the DMDS and DGLL layouts using groups learned by

clustering still outperform competing methods in temporal cost, and only CCDR, which uses the known group information, outperforms DGLL without group information in terms of centroid cost. The DMDS methods also converge more quickly than both Visone and SoNIA.

6 Conclusions

In this paper we proposed a regularized graph layout framework for dynamic network visualization. The proposed framework incorporates grouping and temporal regularization into graph layout in order to discourage nodes from deviating too far from other nodes in the same group and from their previous position, respectively. The layouts are generated in an on-line manner using only present and past data. We introduced two dynamic layout algorithms, DMDS and DGLL, which are regularized versions of their static counterparts, and demonstrated the importance of the regularizers for preserving the mental map in multiple experiments. The proposed methods generalize existing approaches for temporal regularization in MDS and grouping regularization in GLL.

An important area for future work concerns visualization of extremely large dynamic networks containing upwards of thousands of nodes. One issue is related to scalability; both DMDS and DGLL require $O(n^3)$ computational time and thus may not be applicable to extremely large networks. An even more significant issue involves the interpretation of visualizations of such large networks. Even when equipped with temporal and grouping regularization, layouts of extremely large networks may be confusing for a human to interpret so additional techniques may be necessary to deal with this challenge.

A DGLL solution in 2-D

We derive the expressions for ∇f, g, H, and J in 2-D. These vectors and matrices are computed at each iteration in the DGLL algorithm to solve (25) using the interior-point algorithm (Byrd et al., 1999) as discussed in Section 3.3. The constraints can be written as $g(\tilde{X}) = 0$ where

$$g(\tilde{X}) = \begin{bmatrix} \tilde{\mathbf{x}}_1^T M \tilde{\mathbf{x}}_1 - \mathrm{tr}(\tilde{D}) \\ \tilde{\mathbf{x}}_2^T M \tilde{\mathbf{x}}_2 - \mathrm{tr}(\tilde{D}) \\ \tilde{\mathbf{x}}_2^T M \tilde{\mathbf{x}}_1 \end{bmatrix}.$$

The gradient of the objective function is given by

$$\nabla f(\tilde{X}) = \begin{bmatrix} (2\tilde{L} + 2\beta\tilde{E})\tilde{\mathbf{x}}_1 - 2\beta\tilde{E}\tilde{\mathbf{x}}_1[t-1] \\ (2\tilde{L} + 2\beta\tilde{E})\tilde{\mathbf{x}}_2 - 2\beta\tilde{E}\tilde{\mathbf{x}}_2[t-1] \end{bmatrix}.$$

The Jacobian of the constraints is given by

$$J(\tilde{X}) = \begin{bmatrix} 2\tilde{\mathbf{x}}_1^T M & \mathbf{0} \\ \mathbf{0} & 2\tilde{\mathbf{x}}_2^T M \\ \tilde{\mathbf{x}}_2^T M & \tilde{\mathbf{x}}_1^T M \end{bmatrix}.$$

Finally, the Hessian is obtained by

$$\begin{aligned} H(\tilde{X}, \boldsymbol{\mu}) &= \nabla^2 f(\tilde{X}) + \mu_1 \nabla^2 g_1(\tilde{X}) + \mu_2 \nabla^2 g_2(\tilde{X}) + \mu_3 \nabla^2 g_3(\tilde{X}) \\ &= \begin{bmatrix} 2\tilde{L} + 2\beta\tilde{E} + 2\mu_1 M & \mu_3 M \\ \mu_3 M & 2\tilde{L} + 2\beta\tilde{E} + 2\mu_2 M \end{bmatrix}. \end{aligned}$$

B AFFECT evolutionary clustering algorithm

Evolutionary clustering algorithms are designed to cluster dynamic data where a set of objects is observed over multiple time steps. In the dynamic network setting, objects correspond to nodes, and observations correspond to graph adjacency matrices $W[t]$. The AFFECT (Adaptive Forgetting Factor for Evolutionary Clustering and Tracking) framework (Xu et al., 2011a) involves creating a *smoothed adjacency matrix* at each time step and then performing ordinary static clustering on this matrix. The smoothed adjacency matrix is given by

$$\hat{\Psi}[t] = \alpha[t]\hat{\Psi}[t-1] + (1 - \alpha[t])W[t],$$

where $\alpha[t]$ is a forgetting factor that controls how quickly previous adjacency matrices are forgotten. We drop the time index for quantities at time t for simplicity. The AFFECT framework adaptively estimates the optimal amount of smoothing to apply at each time step in order to minimize mean-squared error (MSE) in terms of the Frobenius norm $\mathrm{E}[\|\hat{\Psi} - \Psi\|_F^2]$, where Ψ denotes the expected adjacency matrix $\mathrm{E}[W]$, which can be viewed as a matrix of unknown states characterizing the network structure at time t. It is shown in Xu et al. (2011a) that the optimal choice of α is given by

$$\alpha^* = \frac{\sum_{i=1}^n \sum_{j=1}^n \mathrm{var}\,(w_{ij})}{\sum_{i=1}^n \sum_{j=1}^n \left\{ \left(\hat{\psi}_{ij}[t-1] - \psi_{ij} \right)^2 + \mathrm{var}\,(w_{ij}) \right\}}.$$

For real networks, ψ_{ij} and $\mathrm{var}(w_{ij})$ are unknown so α^* cannot be computed.

The AFFECT framework iteratively estimates the optimal forgetting factor and clusters nodes using a two-step procedure. Begin by initializing the clusters; for example, using the clusters at the previous time step. α^* is estimated by replacing

the unknown means and variances with sample means and variances computed over the clusters. Static clustering is then performed on the smoothed adjacency matrix $\hat{\Psi}$ to obtain cluster memberships. The estimate of α^* and the cluster memberships are then refined by iterating the two steps. In this paper, we use normalized cut spectral clustering (Ng et al., 2001) as the static clustering algorithm. We refer interested readers to Xu et al. (2011a) for additional details.

Acknowledgements

This work was partially supported by the National Science Foundation grant CCF 0830490. Kevin Xu was partially supported by an award from the Natural Sciences and Engineering Research Council of Canada.

References

M. Baur and T. Schank. Dynamic graph drawing in Visone. Technical report, 2008.

M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: Theory and algorithms*. Wiley, 2006.

M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computat.*, 15(6):1373–1396, 2003.

S. Bender-deMoll and D. A. McFarland. The art and science of dynamic network visualization. *J. Social Struct.*, 7(2):1–38, 2006.

S. Bender-deMoll and D. A. McFarland. SoNIA - Social Network Image Animator, 2011. URL `http://www.stanford.edu/group/sonia/`.

I. Borg and P. J. F. Groenen. *Modern multidimensional scaling*. Springer, 2nd edition, 2005.

U. Brandes and S. R. Corman. Visual unrolling of network evolution and the analysis of dynamic discourse. *Inform. Visualiz.*, 2(1):40–50, 2003.

Ulrik Brandes and Dorothea Wagner. A Bayesian paradigm for dynamic graph layout. In *Graph Drawing*, 1997.

Ulrik Brandes and Dorothea Wagner. visone - Analysis and visualization of social networks. In *Graph drawing software*, pages 321–340. 2004.

Ulrik Brandes, Daniel Fleischer, and Thomas Puppe. Dynamic spectral layout of small worlds. In *Graph Drawing*, 2006.

Jürgen Branke. Dynamic graph drawing. In *Drawing graphs: Methods and models*, pages 228–246. 2001.

R. H. Byrd, M. E. Hribar, and J. Nocedal. An interior point algorithm for large-scale nonlinear programming. *SIAM J. Optimiz.*, 9(4):877–900, 1999.

Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng. On evolutionary spectral clustering. *ACM Trans. Knowl. Discov. Data*, 3(4):17, 2009.

J. A. Costa and A. O. Hero III. Classification constrained dimensionality reduction. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2005.

J. de Leeuw and W. J. Heiser. Multidimensional scaling with restrictions on the configuration. In *Multivariate analysis*, pages 501–522. 1980.

G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph drawing: Algorithms for the visualization of graphs*. Prentice Hall, 1999.

N. Eagle, A. Pentland, and D. Lazer. Inferring friendship network structure by using mobile phone data. *Proc. Nat. Acad. Sci.*, 106(36):15274–15278, 2009.

C. Erten, P. J. Harding, S. G. Kobourov, K. Wampler, and G. Yee. Exploring the computing literature using temporal graph visualization. In *Proc. Conf. Visualiz. Data Anal.*, 2004.

Y. Frishman and A. Tal. Online dynamic graph drawing. *IEEE Trans. Visualiz. Comput. Graphics*, 14(4):727–740, 2008.

T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.

E. R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *Graph Drawing*, 2004.

K. M. Hall. An r-dimensional quadratic placement algorithm. *Manage. Sci.*, 17(3): 219–229, 1970.

I. Herman, G. Melançon, and M. S. Marshall. Graph visualisation and navigation in information visualisation: A survey. *IEEE Trans. Visualiz. Comput. Graphics*, 6(1):24–43, 2000.

A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

P. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

Mao Lin Huang and Peter Eades. A fully animated interactive system for clustering and navigating huge graphs. In *Graph Drawing*, 1998.

T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inform. Process. Lett.*, 31(12):7–15, 1989.

Y. Koren. On spectral graph drawing. In *Proc. 9th Int. Comput. Combinat. Conf.*, 2003.

G. Kossinets and D. J. Watts. Empirical analysis of an evolving social network. *Science*, 311(5757):88–90, 2006.

J. A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer, 2007.

J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 1(1):2, 2007.

L. Leydesdorff and T. Schank. Dynamic animations of journal maps: Indicators of structural changes and interdisciplinary developments. *J. American Soc. Inform. Sci. Techol.*, 59(11):1810–1818, 2008.

H. Lütkepohl. *Handbook of matrices*. Wiley, 1997.

K. Misue, P. Eades, W. Lai, and K. Sugiyama. Layout adjustment and the mental map. *J. Visual Lang. Comput.*, 6(2):183–210, 1995.

MIT academic calendar, 2005. URL `http://web.mit.edu/registrar/www/calendar0405.html`.

J. Moody, D. McFarland, and S. Bender-deMoll. Dynamic network visualization. *American J. Sociol.*, 110(4):1206–1241, 2005.

P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J. P. Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, 328 (5980):876–878, 2010.

T. M. Newcomb. *The acquaintance process*. Holt, Rinehart and Winston, 1961.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Proc. Adv. Neural Inform. Process. Systems 14*, 2001.

P. G. Nordlie. *A longitudinal study of interpersonal attraction in a natural group setting*. PhD thesis, University of Michigan, 1958.

J. Sun, Y. Xie, H. Zhang, and C. Faloutsos. Less is more: Compact matrix decomposition for large sparse graphs. In *Proc. 7th SIAM Conf. Data Mining*, 2007.

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal Statist. Soc. Ser. B (Stat. Meth.)*, 58(1):267–288, 1996.

H. Tong, S. Papadimitriou, J. Sun, P. S. Yu, and C. Faloutsos. Colibri: fast mining of large static and dynamic graphs. In *Proc. 14th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2008.

L. N. Trefethen and D. Bau III. *Numerical linear algebra*. SIAM, 1997.

Visone, 2011. URL `http://visone.info/`.

Xiaobo Wang and Isao Miyamoto. Generating customized layouts. In *Graph Drawing*, 1996.

D. M. Witten and R. Tibshirani. Supervised multidimensional scaling for visualization, classification, and bipartite ranking. *Computat. Statist. Data Anal.*, 55 (1):789–801, 2011.

D. M. Witten, R. Tibshirani, and T. Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatist.*, 10(3):515–534, 2009.

K. S. Xu, M. Kliger, and A. O. Hero III. Adaptive evolutionary clustering. *Submitted*, 2011a. URL `http://arxiv.org/abs/1104.1990`.

K. S. Xu, M. Kliger, and A. O. Hero III. Visualizing the temporal evolution of dynamic networks. In *Proc. 9th Workshop Mining Learn. Graphs*, 2011b.

K. S. Xu, M. Kliger, and A. O. Hero III. A regularized graph layout framework for dynamic network visualization: Supporting website, 2012. URL `http://tbayes.eecs.umich.edu/xukevin/visualization_dmkd_2012`.